APPLYING BCMP MULTI-CLASS QUEUEING NETWORKS FOR THE PERFORMANCE EVALUATION OF HIERARCHICAL AND MODULAR SOFTWARE SYSTEMS

S.Balsamo
Università Ca' Foscari di Venezia, Dipartimento di Informatica
Via Torino 155, Venezia
email: `balsamo@dsi.unive.it`

G. Dei Rossi
Università Ca' Foscari di Venezia, Dipartimento di Informatica
Via Torino 155, Venezia
email: `deirossi@dsi.unive.it`

A. Marin
Università Ca' Foscari di Venezia, Dipartimento di Informatica
Via Torino 155, Venezia
email: `marin@dsi.unive.it`

KEYWORDS

Performance evaluation, Software engineering, Queueing networks, Product-form solutions, BCMP

ABSTRACT

Queueing networks with multiple classes of customers play a fundamental role for evaluating the performance of both software and hardware architectures. The main strength of product-form models, in particular of BCMP queueing networks, is that they combine a flexible formalism with efficient analysis techniques and solution algorithms. In this paper we provide an algorithm that starting from a high-level description of a system, and from the definition of its components in terms of interacting sub-systems, computes a multiple-class and multiple-chain BCMP queueing network. We believe that the strength of this approach is twofold. First, the modeller deals with simplified models, which are defined in a modular and hierarchical way. Hence, we can carry on sensitivity analysis that may easily include structural changes (and not only on the time parameters). Second, maintaining the product-form property allows one to derive the average system performance indices very efficiently. The paper also discusses the application of the algorithm for the performance evaluation of Web Sites with modular architectures, such as those based on Content Management Systems.

INTRODUCTION

Performance analysis of modular and hierarchical systems has always been an important topic for the performance evaluation and software engineering research communities (see, e.g., Smith (1990)). In particular, a good approach to software design requires the definition of a modular and hierarchical architecture. From a high-level point of view, the software may be seen as the interaction of several black-box components. The definition of these sub-components follows the same approach in a hierarchical fashion until the very low-level layer of the architecture is reached. Performance evaluation of such models is important since the earlier stages of development as shown in Smith and Williams (2006). In this context, the main problem consists in the definition of efficient algorithms capable of deriving the required performance indices efficiently.

The class of models we consider in this paper is the well-known class of Markovian models. In particular we focus on those models whose underlying stochastic process is a Continuous Time Markov Chain (CTMC). Particular attention will be devoted to BCMP queueing networks introduced in Baskett et al. (1975), i.e., a class of queueing networks with separable solution and for which efficient analysis algorithms have been introduced for instance in Buzen (1973), Resiser and Lavenberg (1980), Bruell et al. (1984), Conway and Georganas (1986), Conway et al. (1989). One of the main features of BCMP queueing networks is the possibility of characterising the customers of the system by assigning them a class (temporary characterisation) and a chain (permanent characterisation). Under a set of assumptions, the class and the chain of a customer determines its probabilistic routing among the queueing stations and the service time distributions.

In this paper we propose a methodology, supported by a novel algorithm, which aims to simplify the performance evaluation of systems designed according to a hierarchical and modular architecture. This methodology is based on the definition of a high level model

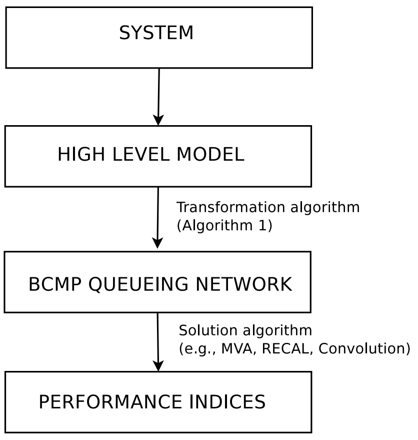

Figure 1: Sketch of the methodology proposed for performance evaluation of modular and hierarchical systems.

of the system consisting of several components. Each of these may be further specified in a hierarchical fashion. Under some assumptions that will be detailed later, we provide an algorithm which transforms this abstract model into a BCMP queueing network. Figure 1 illustrates the steps of this analysis. Let us consider an example. Modern Web Sites are often built on Content Management System (CMS) applications. CMSs are flexible re-programmable software systems consisting of a set of modules that are specialised in some task, e.g., rendering the web page, forum or wiki management, news and comments, user management. Modules are programmed by communities of developers who often work autonomously and must respect the interface given by the core system. Examples of modern CMSs are Drupal, Joomla!, PostNuke, Typo3 just to mention a few. Users who visit the web sites based on CMS are usually not aware of such a modular architecture. Nevertheless, a log analysis may reveal their behaviour among the site modules and clustering techniques may be adopted to distinguish user habits. We aim to provide a modelling approach that allows the system administrator to predict the performance of its web portal under different scenario from the knowledge of:

- the customer behaviours among the modules
- the resource requirements of each module
- the mapping of a required resource to a physical device.

This kind of analysis is not trivial and a hierarchical and modular approach should be adopted as observed for instance in Smith (1990). On the other hand we aim to provide a methodology that is compatible with known exact analysis algorithms to avoid the need of simulation or approximated technique as usually done in Woodside et al. (1995).

The main contribution of this paper consists in providing a methodology supported by an original algorithm that allows the modeller to specify the system in terms of a multiple-class and multiple-chain queueing network (QN) in which each station is itself a multiple-class and multiple-chain QN. The peculiarity of this hierarchical approach is that each station at a given level of abstraction is defined in isolation but, given two or more stations, they may share one of their components at a lower level of abstraction. In the example of the CMS one may think that at the top level of the CMS one has the routing of customers among the site modules (stations at the top level). Each module is then defined in terms of usage of resources (e.g., database, CPU, etc.). However, when these modules are combined one should be able to specify whether a resource is shared among different modules or the modules have distinct resources available. Obviously, this may have great impact on the overall performance of the system (e.g., are the DB and the multimedia resources stored in the same hard disk?). The goal of the algorithm that we introduce is to transform a QN defined at a top level into one defined at a lower level until the lowest level is reached. Once this is done, under some assumptions that will be described in the following sections, we obtain a product-form BCMP QN that may be analysed by the well-know algorithms for the computation of the average performance indices in steady-state.

The paper is structured as follows. First, we briefly recall the theoretical background on multiple-class BCMP queueing networks. Then, we describe the proposed methodology and we define of the algorithm. The last section provides an application example of the proposed approach. Some final remarks conclude the paper.

THEORETICAL BACKGROUND

This section aims to briefly recall the fundamental theorem on multiple-class product-form queueing networks, i.e., the BCMP theorem (Baskett et al. (1975)). Informally, we can say that it states sufficient conditions for a QN with multiple classes of customers to yield a product-form solution. Its importance is not only theoretical because several algorithms have been defined to compute the average performance indices in steady-state efficiently (e.g., Buzen (1973), Resiser and Lavenberg (1980), Conway and Georganas (1986)). This section first briefly illustrates the BCMP theorem and then lists the algorithms for the analysis with their computational complexity.

The BCMP theorem

BCMP queueing networks consist of a set of queueing centers and a (possibly infinite) set of customers. At a given epoch, each customer in the network has a class which may determine its routing probabilities or the service time distribution at a given service station. When a customer changes its class we talk about *class switching.* Note that, in this paper, we use the concept of class in a local sense as in Chandy and Sauer (1980) rather in the global one used in Baskett et al. (1975). Classes form a temporary partition of the customers while chains are a permanent partition. Each class of customers belongs to a chain and routing may occur only within the same chain. Some conditions on the probabilistic routing must be assumed in order to ensure the ergodicity of the underlying process (see Balsamo and Marin (2007) for a recent survey). A chain may be open or closed. In the former case, customers arrive from the outside according to a Poisson process with a given rate, while in the latter the number of customers for that chain must be specified. The network is called *open* if all its chains are open, *closed* if they are all closed or *mixed* otherwise. Queueing stations must belong to one of the following types:

Type 1 : The queueing discipline is First Come First Served (FCFS) and the service time distribution is exponential and class-independent,

Type 2 : The queueing discipline is Processor Sharing (PS),

Type 3 : The station has infinite servers (IS), hence customers never waits in queue (*Delay Stations*),

Type 4 : The service discipline is Last Come First Served with Preemptive Resume (LCFSPR).

Stations of type 2, 3 or 4 may have a Coxian distributed service time that depends on the customer class. Moreover, the station service time may depend on the queue length at a given epoch (some non-strict conditions must be satisfied). This allows one to model important features such as the effect of multiple servers in the same station. Table 1 illustrates the notation we adopt and that we now briefly summarise. We use $\Omega = \{S_1, \dots, S_M\}$ to denote the set of M queueing stations of the network, and let $\mathcal{R}_i$ be the set of classes served by station S_i, with R_i elements that are usually denoted by letters $r, s, \dots$. Let $\mathcal{C} = \{1, \dots, C\}$ be the set of labels for the C chains of the QN, then $\mathcal{R}_i^{(c)}$ is the set of classes served by station S_i and belonging to chain c (with $R_i^{(c)}$ elements), $1 \le c \le C$ and $1 \le i \le M$. Clearly, $\cup_{c=1}^{C} \mathcal{R}_i^{(c)} = \mathcal{R}_i$ for all i. The state-independent probabilistic routing is described by the probability matrix $\mathbf{P}^{(c)}$ for each chain c. Elements of $\mathbf{P}^{(c)}$ are $p_{ri,sj}^{(c)} \ge 0$, with $1 \le i, j \le M$ and $r \in \mathcal{R}_i^{(c)}$, $s \in \mathcal{R}_j^{(c)}$ and represent the probability of a customer entering station S_j with class s after being served in station S_i as class r. Label 0 represents the outside (hence matrix $\mathbf{P}$ has $1 + \sum_{i=1}^{M} R_i$ rows and columns). Sometimes, we have just one routing matrix $\mathbf{P}$ and we desire to derive the partition in $\mathbf{P}^{(c)}$, i.e., identify the chains in the QN. This can be reduced to the problem of identifying the ergodic sub-components in a Markov Chains and, since the structure of the network is usually rather small, the problem is known to be computationally tractable (see, e.g., Kant (1992), Balsamo and Marin (2007)). If c, $1 \le c \le C$, is a closed chain, then $K^{(c)}$ denotes the number of customers and $p_{ri,0}^{(c)} = p_{0,ri}^{(c)} = 0$ for all $r \in \mathcal{R}_i^{(c)}$ and $i = 1, \dots, M$. If c is open $\lambda^{(c)}$ is the total arrival rate and matrix $\mathbf{P}^{(c)}$ is such that element $p_{0,ri}^{(c)}$ is the probability that a customer arriving from the outside enters station i as class r and element $p_{ri,0}^{(c)}$ is the probability for a customer to leave the system after being served at station i with class r. Before briefly stating the BCMP theorem, we recall the definition of the QN traffic equations. For an open chain c, the system of traffic equations are:

$$\mathbf{e}_{ri}^{(c)} = \lambda_{ri}^{(c)} p_{0,ri}^{(c)} + \sum_{j=1}^{M} \sum_{s \in \mathcal{R}_j^{(c)}} e_{sj}^{(c)} p_{sj,ri}^{(c)} \tag{1}$$

for all $i = 1, \dots, M$ and $r \in \mathcal{R}_i^{(c)}$. If c is a closed chain, the corresponding system of traffic equations is:

$$\mathbf{e}_{ri}^{(c)} = \sum_{j=1}^{M} \sum_{s \in \mathcal{R}_j^{(c)}} e_{sj}^{(c)} p_{sj,ri}^{(c)} \tag{2}$$

for all $i = 1, \dots, M$ and $r \in \mathcal{R}_i^{(c)}$. In the latter case the system is under-determined, and the solution is defined up to an arbitrary non-null constant that has to be chosen. Solutions $e_{ri}^{(c)}$ of systems (1) and (2) represent the (relative) visit ration to station i, class r of chain c. Vector $\mathbf{e}_i = (e_{ri})$ with $r \in \mathcal{R}_i$ plays a pivotal role for the network steady-state solution. We can now state the salient result of the BCMP theorem given in Baskett et al. (1975).

Theorem 1 (BCMP (salient results)) *Let us consider a multiple-class and multiple-chain QN, open, closed or mixed, whose queueing stations are of type* 1, 2, 3 *or* 4. *Then, if the underlying stochastic process is ergodic, the steady-state probabilities are in product-form with respect to the queueing stations, i.e., let* $\mathbf{n} = (n_1, \dots, n_M)$ *be the vector representing the state of the network, where component* n_i *is the state of station* S_i*, then the following relation holds:*

$$\pi(\mathbf{n}) = \frac{1}{G} \prod_{i=1}^{M} g_i(n_i), \tag{3}$$

where π *is the steady-state distribution of the QN, and* $g_i(n_i)$ *is the steady-state distribution of station* S_i *considered in isolation, with arrival rates* $\mathbf{e}_i$*, and* G *is a normalising constant.*

Ω	Set of queueing stations of the network
$\lambda^{(c)}$	Arrival rate to open chain c
$\mathcal{C}$	Set of the chain labels
C	Number of chains
$e_{ri}^{(c)}$	(Relative) arrival rate to station S_i, class r of chain c
$K^{(c)}$	Population of chain c
$\mathbf{e}_i$	Solution for the traffic equation systems (1) or (2) of station S_i
M	Number of queueing stations
$\mathbf{P}^{(c)}$	Routing probability matrix for chain c
$p_{ri,sj}^{(c)}$	probability for a customer to enter station S_j, class s, after being served at station S_i, class r, where both the classes s and r belong to chain c
$\mathcal{R}_i^{(c)}$	Set of classes of chain c served by station S_i
$\mathcal{R}_i$	Set of classes served by station S_i
$R_i^{(c)}$	Number of classes served by station S_i belonging to class c
R_i	Number of classes served by station S_i

Table 1: Table of the notation.

Solution algorithms

Theorem 1 and the class of BCMP networks have been widely applied for system performance analysis, because several efficient solution algorithms have been defined to compute the stationary state distribution π and a set of average performance indices. Such algorithms specifically apply to analyse closed or mixed networks, where we have to compute the normalising constant G, as stated by Theorem 1. Note that for open networks we have $G = 1$, the solution $e_{ri}^{(c)}$ of the traffic equations (1) already gives the throughputs of each node S_i for classes r in chain c. Then one can easily derive the other average performance indices by classical queueing system results.

Various solution algorithms have been defined for closed and mixed BCMP networks. Some algorithms, such as the Convolution Algorithm, directly compute the normalising constant G in equation (3) and hence a set of mean performance indices, such as the mean response time, the average queue length, and the throughput of each queueing station. Other algorithm, such as MVA (Mean Value Analysis) avoid the computation of the normalising constant G and iteratively (over the number of customers) directly compute a set of average performance indices. For multiple-class and multiple-chain BCMP networks some algorithms apply special recursive scheme on the number of chains, and/or take advantage of the possible sparsity of the chains (e.g., chains that contain few classes) to derive efficient solution. Although it is out of the scope of this paper to describe these well-known algorithms, we just cite them and recall their computational complexity. Several tools for the analysis of queuing networks have been implemented over the last decades. A recent work, called qnetworks toolbox, is described in Marzolla (2010). Such an implementation of several algorithms is given in terms of library of functions for Octave, i.e., a programmable environment for numerical computation. This allows one to integrate easily the algorithms of qnetworks with new ones, for instance that presented by Algorithm 1. Hereafter, we consider a queueing network with multiple chains but where each station has just one class per chain (single-class, multiple-chain QN). One can show that for each multiple-class and multiple-chain BCMP QN it is possible to define another BCMP QN with single-class and multiple-chain with the same average performance indices (see, e.g., Kant (1992)). For the sake of clarity, we consider the QN consisting of only closed chains.

The Convolution Algorithm computes the normalising constant from which the average performance indices may be derived. The computational complexity, given the solution of the traffic equations system (2), depends on the type of stations in the QN. In particular each iteration has a cost of $\mathcal{O}(CH)$ for load-independent stations and of $\mathcal{O}(H^2)$ for the others, where $H = \prod_{c=1}^{C}(K^{(c)}+1)$. If all the chains has the same population κ and no load-dependent stations are present, then the computational cost is $\mathcal{O}(MC\kappa^C)$.

The Mean Value Analysis algorithm (MVA) is based on the *Arrival theorem* that provides an efficient recursive scheme to compute the steady-state average performance indices. For a QN without load-dependent stations, and with identical chain populations, its complexity is identical to that of the Convolution.

The Recursion by Chain Algorithm (RECAL), defined in Conway and Georganas (1986), computes the normalising constant and, in a similar fashion of Convolution, from this it derives the average performance indices. It is particularly interesting because despite of a greater complexity in the implementation, its computational complexity grows in a polynomial way with the number of chains, i.e., for high number of chains,

$\mathcal{O}(C^{M+1})$. RECAL has been improved from its original definition in several ways and is now widely applied for the solution of QNs with high number of chains.
Note that several other algorithms for the exact or approximate computation of the average performance indices in multiple-chain BCMP QNs have been defined in literature. A survey may be found in Balsamo and Marin (2007).

FRAMEWORK DESCRIPTION AND ALGORITHM DEFINITION

In this section we first illustrate how to describe a model in our framework, and then we present the algorithm to obtain the underlying BCMP QN. Once this is derived one of the algorithms presented in the previous section may be applied in order to obtain the desired performance indices.

Model description

As we pointed out in the introduction we aim to provide a framework for the specification of software and hardware architectures which enhances the modularity and hierarchical features. In this setting, we see a system, at its highest level of abstraction, as consisting of a set of components $d_1, d_2, \ldots, d_{\ell_1}$. The easiest way to interpret the model specification is seeing these components as the queueing stations of a multiple-class and multiple-chain QN. Hence, probabilistic routing and customer characterisations are allowed. Each of the components d_i, with $d_i = d_1, \ldots, d_{\ell_1}$, seen at the highest level of abstraction, may be defined as:

- A BCMP queueing station
- A sub-model consisting of components $d_{(i)1}, \ldots, d_{(i)\ell_2}$. Note that it is *not* the case that $d_{(i)k} \neq d_j$ for all $1 \leq j \leq \ell_1$ and $1 \leq k \leq \ell_2$, i.e., a component may use a resource which has already been described at a higher level. These components interact as stations of an open multiple-class and multiple-chain QN. Each sub-model from the outside can be seen as a black box, with a set of access points with some labels, i.e., the classes of the customers arriving from the outside (input classes) and the classes of the customers leaving the sub-model (output classes). We require that the set of input classes must be equal to the set of output classes of each component.

This recursive definition is the basis of the algorithm that follows.
We now introduce the concept of *well-formed* model.

Definition 1 (Well-formed models) *Given a model consisting of m components (in any level of abstraction) then we define the binary relation $\succ$ as follows:*

- *$d \succ d'$ if and only if d' appear in the definition of d*

and the binary relation $>$ as follows:

- *$d \succ d' \Rightarrow d > d'$ or*
- *$d > d'$ if there exists d'' such that $d \succ d''$ and $d'' > d'$.*

A model is well-formed if and only if relation $>$ is a strict partial order.

Roughly speaking, a well-formed model does not have cycles in the definition of the components. However, it is possible, at a given level of abstraction, to refer to components specified at higher levels. Hereafter, we consider only well-formed models.

Algorithm definition

In order to better understand our approach to modular BCMP network design, we will first consider the algorithm, then we show how it could be further optimized.
Let $\mathcal{D}$ be the set of the m components $d_1 \ldots d_m$ that form the model, and $\mathcal{R}_i$ be the set of the n classes $r_{i,1} \ldots r_{i,n}$ for the component d_i. In each component, we call EI (external input) the arrival streams from the outside, and EO (external output) the departure streams.
Binary relation $d \succ d'$ given by Definition 1, means that d *contains* d', i.e., if d is not a simple QN station, then d' appears in the routing matrix of d.
Let $\mathbf{P}_d$ be the routing matrix of component d and let be $\mathbf{P}_{d'}$, with $d \succ d'$, the routing matrix of a subcomponent of d' of d, then we aim to unfold d' in order to specify a routing matrix for its subcomponents.
If $d_i \succ d_j$, let $A_{d_i,d_j} : \mathcal{R}_i \to \mathcal{R}_j$ be a partial function that associates class names of component d_i with class names of component d_j. Notice that A is not necessarily injective, i.e., two or more classes of the container component can be mapped into the same class of the contained component. Whenever it happens, we should add a new class set to the submodel.
$A_{i,j}$ functions must be given by the user of the algorithm, and define how classes of a high level component maps on classes of its subcomponents. This allows the design of a low level component without knowledge of its future use in a high level one.
In Algorithm 1 we show a method for routing matrix unfolding. To keep the code simple, we use a single routing matrix that combines every chain of the component, but the algorithm preserves chains, i.e. (node,class) pairs that are in different chains in high-level model description remain in different chains in the solution computed by the algorithm.
Notice that here the insertion and replacement operators for rows and columns have a loose semantics, i.e., whenever a row or a column with less elements is assigned to

```
Input: routing matrix $\mathbf{P}_d$ of component $d$, component
       counter array $Dc$, functions $A_{i,j}$
Output: unfolded routing matrix $\mathbf{P'}_d$ of component $d$
if $d$ is a station then /* base case */
    foreach class $r$ of $d$ do
        insert in $\mathbf{P'}$ rows and a columns for $EI_d, r$ and
        $EO_d, r$ of $\mathbf{P}$
    end
else
    foreach $d_i | d \succ d_i$ do
        $Dc_i \leftarrow Dc_i + 1$
        Let $Rc_i$ be a class counter array
        foreach class $r_k$ of $d_i$ as named in $d$ do
            $r_{i,j} \leftarrow \mathtt{A}_{d,d_i}(r_k)$
            $Rc_{i,j} \leftarrow Rc_{i,j} + 1$
            if $Rc_{i,j} > \max Rc_i$ then
                $\mathbf{U}$ = UnfoldComponents($\mathbf{P}_{d_i}$)
                rename each class $r_{i,j}$ of $d_i$ in $\mathbf{U}$ as
                $r_{i,j,Dc_i,Rc_{i,j}}$
                insert in $\mathbf{P'}$ rows and columns of $\mathbf{U}$
            end
            replace column $EI_{d_i}, r_{i,j,Dc_i,Rc_{i,j}}$ in $\mathbf{P'}$ with
            column $d_i, r_k$ of $\mathbf{P}$
            replace row $EO_{d_i}, r_{i,j,Dc_i,Rc_{i,j}}$ in $\mathbf{P'}$ with
            row $d_i, r_k$ of $\mathbf{P}$
        end
    end
end
return $P'$
```

Algorithm 1: Algorithm *UnfoldComponents* to derive a BCMP QN from a model specified at a higher level of abstraction.

a greater one, all elements that are not indexed in the smaller vector are set to 0.
The main idea of the algorithm is to distinguish the use of the same component class in different incoming and outgoing path, creating a new class whenever the component or the class itself is reused more than once. In order to achieve this, we use a global component usage counter and a local one for class usage. The algorithm then recursively expand the hierarchical model in a top-down fashion, until it reaches a standard BCMP station, i.e., a sub-model that has no components.

Comments on the algorithm. It is possible to show that starting from a high-level model whose routing is specified correctly, i.e., none of the classes become empty or its population grows indefinitely with probability 1 in the long run, we obtain a valid BCMP QN. Hence, under stability assumptions, the steady-state distribution and the average performance indices may be derived. The strength of the algorithm is that parametric analysis are simplified with respect to using the BCMP model directly. In fact, it suffices to change the labels associated with some resources to generate a totally different

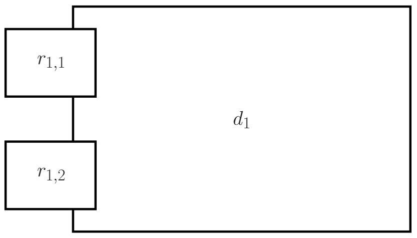

Figure 2: A black-box model of a database-indexed file archive CMS module.

routing. In the former example of the CMS, the administrator may be able to predict the performance measures of its system in case of a new server quite easily by mapping which server modules will be run in the new server at the highest level of abstraction.
The algorithm could be optimized, without changing its behaviour, saving partial computations of the $\mathbf{U}$ matrix before the execution of the innermost foreach cicle and renaming, at each iteration, all classes accordingly.
Under the assumption that, on average, every component has the same number of sub-components d, every component that is not a BCMP queueing station has the same number of classes r and the depth of the model, i.e., the length of the longest chain $d_1 \succ \ldots \succ d_k$, is n, we estimate that, in the worst case, the algorithm complexity is $\mathcal{O}(rd^n)$.

Illustrating example

We now provide an example that aims to illustrate the modelling methodology and the application of Algorithm 1 to a case-study. For the sake of brevity we keep the modelled system rather simple even if, obviously, the algorithm usefulness is enhanced by larger systems.
System description. We consider just a single module of a CMS, i.e., a database-indexed file archive. This is a typical feature of many websites that provide downloadable resources (e.g., multimedia or documents). Suppose that this module serves two classes of customers, one which models the file upload, and the other the file download. A download request passes through the database and then accesses the disk. An upload request, passes through the database and the accesses the disk to be written. If the operation is succesfull then a message for the database is generated to confirm the correct operation.
Model definition. The black-box model of the CMS module is shown in Figure 2. We can see the two incoming and outgoing classes of customers. Let this component name be d_1. Figure 3 represents the internal structure of d_1, where d_2 is the database component and d_3 the disk component. Hence, we clearly have $d_1 \succ d_2$, $d_1 \succ d_3$ and $A_{1,2}(r_{1,1}) = r_{2,1}, A_{1,2}(r_{1,2}) = r_{2,1}, A_{1,2}(r_{1,3}) = r_{2,2}, A_{1,3}(r_{1,1}) = r_{3,1}, A_{1,3}(r_{1,2}) =$

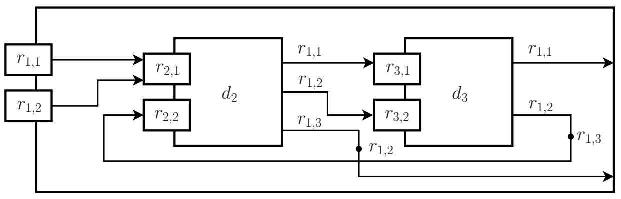

Figure 3: The internal definition of the module of Figure 2.

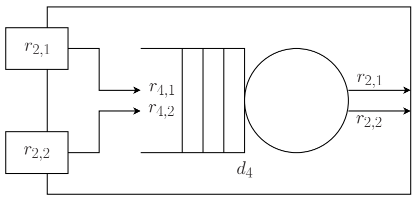

Figure 4: Minimal DB module design.

$r_{3,2}$. The routing matrix $\mathbf{P}_1$ is, ignoring impossible (component,class) combinations, a 9×9 square matrix.

Let us suppose, for the sake of brevity, that component d_2, as in Figure 4, is made of a single component, d_4, that is a multiple-class station, and that $A_{2,4}(r_{2,1}) = r_{4,1}, A_{2,4}(r_{2,2}) = r_{4,2}$. Then we show how an application of the algorithm to d_1 transforms d_2. Note that we limit the observation to this part of the system for the sake of readability, since the number of classes arising from this simple example may be difficult to represent graphically.

The algorithm invokes recursively the UnfoldComponents function of Algorithm 1 until it finds a BCMP queueing station, in this case d_4, then, assuming it is encountered for the first time, it renames its classes $r_{4,1}$ and $r_{4,2}$ in $r_{4,1,1,1}$ and $r_{4,2,1,1}$. At the end of the invocation of UnfoldComponents($\mathbf{P}_{d_2}$), the resulting routing matrix is like in Table 2. Notice that rows $EO_{r_{2,1}}$ and $EO_{r_{2,2}}$ are both zero, because they represents departures from the component, that have yet to be connected to a higher level sub-model.

During the invocation of UnfoldComponents($\mathbf{P}_{d_1}$) both $A(r_{1,1})$ and $A(r_{1,2})$ return $r_{2,1}$, and therefore $Rc_{2,1}$ is incremented twice. The algorithm, then, inserts the elements of $\mathbf{P}_{d_2}$ twice in $\mathbf{P}'$, with different class names, e.g., $r_{4,1,2,1}$ and $r_{2,1,1,2}$.

As previously stated, the usefulness of the algorithm become more noticeable when the model is complex, e.g., if the Database module, instead of being made of a single queueing station, was described in terms of interacting submodels, like a CPU, one or more disks, a caching system, et cetera. All this subsystems may be also used by other modules.

CONCLUSION

This paper addresses the problem of combining a modular and hierarchical modelling technique with an efficient analysis method. The main theoretical contribution is an algorithm which allows the transformation of models defined at a higher level of abstraction into models defined at a lower one. A recursive application of this algorithm produces, under a set of conditions, a product-form multiple-class and multiple-chain BCMP QN. Although to obtain this we must limit the formalism expressivity, e.g., fork and join constructs are not permitted, our aim is to provide a methodology supported by efficient algorithms for the exact analysis. Other more expressive hierarchical approaches, such as those defined in Woodside et al. (1995), may require approximate algorithms for deriving the average performance indices in equilibrium. Future works have several directions. From a theoretical point of view, an extension of the class of models tractable by such a formalism would be important. Another important aspect is the integration of this framework with well-known and recent advances in web mining, particularly in log analysis. These techniques provide a partition of application users according to their behaviour. Although these techniques have been traditionally applied to predict the customer interests (e.g., for proposing context aware advertisements) we claim they may be very useful also for performance evaluation purposes.

REFERENCES

Balsamo S. and Marin A., 2007. *Queueing Networks in Formal methods for performance evaluation*, M. Bernardo and J. Hillston (Eds), LNCS, Springer, chap. 2. 34–82.

Baskett F.; Chandy K.M.; Muntz R.R.; and Palacios F.G., 1975. *Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. J ACM*, 22, no. 2, 248–260.

Bruell S.C.; Balbo G.; and Afshari P.V., 1984. *Mean Value Analysis of Mixed, Multiple Class BCMP Networks with Load Dependent Service Stations. Perform Eval Elsevier*, 4, 241–260.

Buzen J.P., 1973. *Computational algorithms for closed queueing networks with exponential servers. Commun ACM*, 16, no. 9, 527–531.

Chandy K.M. and Sauer C.H., 1980. *Computational algorithms for product form queueing networks. Commun ACM*, 23, no. 10, 573–583.

Conway A.E.; de Souza e Silva E.; and Lavenberg S.S., 1989. *Mean Value Analysis by Chain of Product form Queueing Networks. IEEE Trans Comput*, 38, no. 3, 432–442.

$$\mathbf{P}' = \left[\begin{array}{c|cccccc}
 & EI, r_{2,1} & EI, r_{2,2} & EO, r_{2,1} & EO, r_{2,2} & d_4, r_{4,1,1,1} & d_4, r_{4,2,1,1} \\
\hline
EI, r_{2,1} & 0 & 0 & 0 & 0 & 1 & 0 \\
EI, r_{2,2} & 0 & 0 & 0 & 0 & 0 & 1 \\
EO, r_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 \\
EO, r_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 \\
d_4, r_{4,1,1,1} & 0 & 0 & 1 & 0 & 0 & 0 \\
d_4, r_{4,2,1,1} & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right]$$

Table 2: Routing table $\mathbf{P}'$ at the end of UnfoldComponents($\mathbf{P}_{d_2}$).

Conway A.E. and Georganas N.D., 1986. *RECAL - a new efficient algorithm for the exact analysis of multiple-chain closed queuing networks*. *J ACM*, 33, no. 4, 768–791.

Kant K., 1992. *Introduction to Computer System Performance Evaluation*. McGraw-Hill.

Marzolla M., 2010. *The qnetworks Toolbox: A Software Package for Queueing Networks Analysis*. In *Proc of Int. Conf. ASMTA*. Cardiff, UK, *LNCS*, vol. 6148, 102–116.

Resiser M. and Lavenberg S.S., 1980. *Mean Value Analysis of Closed Multichain Queueing Network*. *J ACM*, 27, no. 2, 313–320.

Smith C.U., 1990. *Performance Engineering of Software Systems*. Addison-Wesley.

Smith C.U. and Williams L.G., 2006. *Five steps to establish software performance engineering*. In *CMG Conf.* Reno, Nevada, USA, 507–516.

Woodside C.; Neilson J.; Petriu S.; and Mjumdar S., 1995. *The Stochastic rendezvous network model for performance of synchronous client-server-like distributed software*. *IEEE Transaction on Computer*, 44, 20–34.

BIOGRAPHY

SIMONETTA BALSAMO is a full professor of Computer Science at the University Ca' Foscari of Venice, Italy. Her research interests include performance and reliability modeling and analysis of computer and communication systems, parallel and distributed processing, distributed simulation, quantitative analysis of software architectures and integration of specification languages and performance models. She has published several papers in international journals, conference proceedings, books and special editions. has servedd as general chair, program chair and program committee member for several internatinal conferences, is associated editor of Performance Evaluation Journal.

GIAN-LUCA DEI ROSSI received his M.Sc. degree in Computer Science from the University of Venice in 2010. He is currently a Ph.D. student at the same University. His research area is on stochastic models and performance evaluation.

ANDREA MARIN received his degree in Computer Science from the University of Venice in 2002, and the Ph.D. in Computer Science from the same University in 2009. Most of his research is focused on the analysis of compositionality of Markovian stochastic models, in particular in stochastic Petri net domain. Some novel results about product-form solutions have been derived in this context. He now works as a post-doctoral researcher at the University of Venice